Exp No: 7

Object Detection

Srinidhi.V
18CL02

Aim:

To implement object detection using convolutional neural networks for MNIST semi-supervised Image Recognition.

Algorithm:

1. Start
2. Develop a Baseline model for the MNIST dataset.
   2.1 First load the dataset using mnist.load_data().
   2.2 We reshape the data and then we use a one hot encoding for the class using the categorical() utility function.
   2.3 Next we prepare the pixel data by typecasting them from into float and then normalise the pixels.
3. Defining model
   3.1 We define the model using a single convolutional layer, followed by a max pooling layer.
   3.2 All the layers will use a RELU activation function
4. Then we evaluate the model using five fold cross validation, and we present the results using a line chart and a box and whiskers plot.

5. The next step is to improve the conducted model, by using Batch normalization. We can further improve the model by increasing the depth of the model using VGG-like pattern.

6. After finalising our model, we fit and save the final model & then we evaluate it.

7. Finally a sample image is loaded and model is tested for accurate predictions.

8. Stop.

Sample Input And Output:

The digits image dataset is loaded as input. The output will display the Cross Entropy Loss and classification Accuracy and the mean, Standard Deviation.

Accuracy: mean = 98.643

std = 0.044.

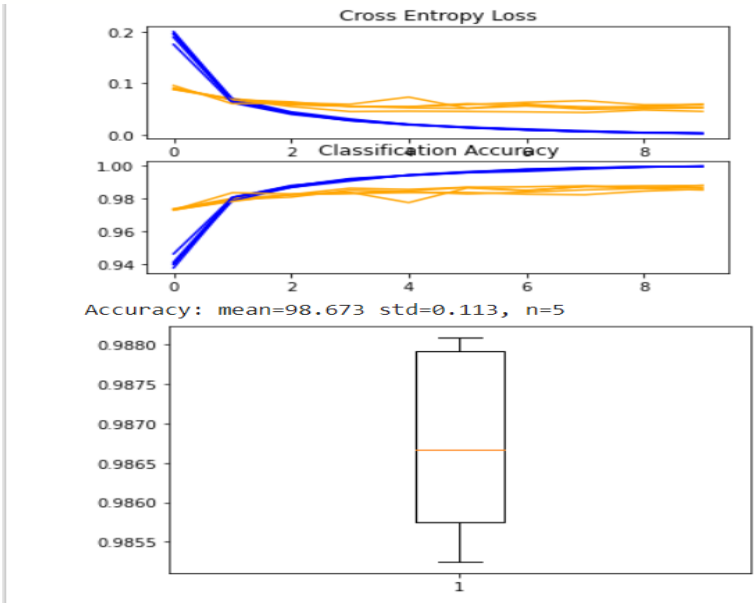
Program:

```
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD

def load_dataset():
  (trainX, trainY), (testX, testY) = mnist.load_data()
  trainX = trainX.reshape((trainX.shape[0], 28, 28, 1))
  testX = testX.reshape((testX.shape[0], 28, 28, 1))
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY

def prep_pixels(train, test):
  train_norm = train.astype('float32')
  test_norm = test.astype('float32')
  train_norm = train_norm / 255.0
  test_norm = test_norm / 255.0
  return train_norm, test_norm

def define_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shap
e=(28, 28, 1)))
  model.add(MaxPooling2D((2, 2)))
  model.add(Flatten())
  model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
  model.add(Dense(10, activation='softmax'))
  opt = SGD(lr=0.01, momentum=0.9)
  model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
  return model

def evaluate_model(dataX, dataY, n_folds=5):
  scores, histories = list(), list()
  kfold = KFold(n_folds, shuffle=True, random_state=1)
  for train_ix, test_ix in kfold.split(dataX):
    model = define_model()
```

```
    trainX, trainY, testX, testY = dataX[train_ix], dataY[train_ix], dataX[test_ix], dataY[test_ix]
    history = model.fit(trainX, trainY, epochs=10, batch_size=32, validation_data=(testX, testY
), verbose=0)
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> %.3f' % (acc * 100.0))
    scores.append(acc)
    histories.append(history)
  return scores, histories

def summarize_diagnostics(histories):
  for i in range(len(histories)):
    pyplot.subplot(2, 1, 1)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(histories[i].history['loss'], color='blue', label='train')
    pyplot.plot(histories[i].history['val_loss'], color='orange', label='test')
    pyplot.subplot(2, 1, 2)
    pyplot.title('Classification Accuracy')
    pyplot.plot(histories[i].history['accuracy'], color='blue', label='train')
    pyplot.plot(histories[i].history['val_accuracy'], color='orange', label='test')
  pyplot.show()

def summarize_performance(scores):
  print('Accuracy: mean=%.3f std=%.3f, n=%d' % (mean(scores)*100, std(scores)*100, len(sc
ores)))
  pyplot.boxplot(scores)
  pyplot.show()

def run_test_harness():
  trainX, trainY, testX, testY = load_dataset()
  trainX, testX = prep_pixels(trainX, testX)
  scores, histories = evaluate_model(trainX, trainY)
  summarize_diagnostics(histories)
  summarize_performance(scores)

run_test_harness()
```

Output:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
> 98.525
> 98.667
> 98.575
> 98.808
> 98.792
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:55: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently reuses the earlier i
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:59: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently reuses the earlier i
```



Result:

Thus we have implemented object detection using convolutional networks for semi-supervised Image Recognition.